

Dynamic Hybrid Traffic Flow Modeling

CISIT Phase 5 — ISART Project

Scientific report 2013

Hassane Abouaïssa, Yoann Kubera, Gildas Morvan

<http://www.lgi2a.univ-artois.fr/~morvan/>
gildas.morvan@univ-artois.fr

Univ Lille Nord de France, F-59000 Lille, France
UArtois, LGI2A, F-62400, Béthune, France



Acknowledgments This work has been supported by International Campus on Safety and Intermodality in Transportation (CISIT), the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the National Center for Scientific Research (CNRS).

Contents

1	Introduction	3
1.1	Context	3
1.2	Motivations	3
2	SIMILAR	4
2.1	Motivations	4
2.2	Related works	4
2.3	SIMILAR micro kernel	5
2.4	SIMILAR architecture	6
2.5	Simulation engine	6
2.6	Probes	7
3	JAM-FREE	8
3.1	Motivations	8
3.2	Related works	8
3.3	JAM-FREE use cases	9
3.4	Road network structure	9
3.5	Vehicle behaviors	11
3.5.1	Acceleration model	12
3.5.2	Lane changing model	12
3.6	Traffic generation	12
3.7	User interfaces	13
4	Conclusion	13
4.1	Deliverables	13
4.2	Perspectives	13
	Appendices	14
A	JAM-FREE XML files	14
B	JAM-FREE Framework	18
C	SonarQube screenshots	18
	References	18

Résumé en français

La simulation du trafic routier sur des réseaux de grande échelle est un problème compliqué car il suppose d'intégrer dans un même modèle différentes approches. Ainsi, les sections autoroutières sont généralement représentées à l'aide de modèles macroscopiques alors que pour les sections urbaines, des modèles microscopiques sont utilisés. De manière générale, les modèles microscopiques sont intéressants lorsque les interactions entre véhicules, ainsi que la topologie du réseau deviennent complexes.

Des modèles intégrant ces différents niveaux de représentation sont généralement qualifiés d'hybrides. Par ailleurs, ils sont généralement "statiques" : à chaque portion du réseau est associée une représentation unique qui ne changera pas au cours de la simulation. Afin de palier cette limitation, Nous avons débuté en 2013 dans le cadre du projet ISART le développement d'un simulateur multi-agent multi-niveaux de flux de trafic routier nommé JAM-FREE permettant :

- de simuler des réseaux routier de grande taille efficacement en utilisant la technique du niveau de détail dynamique,

- de tester de nouveaux algorithmes de régulation, observation et routage.

Ce simulateur repose sur un framework de modélisation et de simulation multi-agents multi-niveaux nommé SIMILAR (SIMulations with Multi-Level Agents and Reactions), implémenté en Java (Morvan and Kubera, 2014a,b), distribué prochainement sous licence libre.

Dans ce rapport, nous présentons ces résultats scientifiques ainsi que les publications associées.

I Introduction

This paper reports the works conducted by Hassane Abouaïssa¹, Yoann Kubera² and Gildas Morvan³ during the phase 5 of CISIT⁴ within the ISART project.

I.1 Context

A flow of moving agents can be observed at different scales. Thus, in traffic modeling, three levels are generally considered: the *micro*, *meso* and *macro* levels, representing respectively the interactions between vehicles, groups of vehicles sharing common properties (such as a common destination or a common localization) and flows of vehicles. Each approach is useful in a given context: micro and meso models allow to simulate road networks with complex topologies such as urban area, while macro models allow to develop control strategies to prevent congestion in highways.

However, to simulate large-scale road networks, it can be interesting to integrate different representations, *e.g.*, micro and macro, in a single model as shown on fig. 1. Some existing hybrid micro-macro traffic models are shown in table 1.

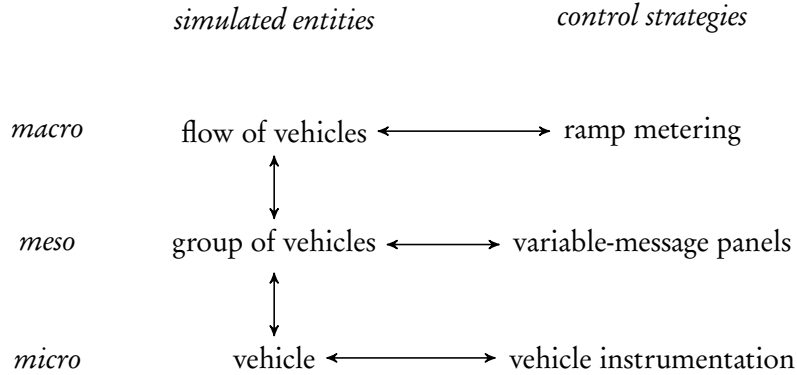


Figure 1: Hybrid traffic simulation and control approach

I.2 Motivations

The models presented in table 1 share the same limitation: connections between levels are fixed *a priori* and cannot be changed at runtime. Therefore, to be able to observe some emerging phenomena such as congestion formation or to find the exact location of a jam in a large macro section, a dynamic hybrid modeling approach is needed (Sewall et al., 2011). Multi-level agent-based modeling is an interesting

¹<http://www.lgi2a.univ-artois.fr/spip/spip.php?article5&idpersonne=5>

²<http://www.yoannkubera.net/>

³<http://www.lgi2a.univ-artois.fr/~morvan/>

⁴<http://www.cisit.org/>

model	micro model	macro model
Magne et al. (2000)	SITRA-B+	SIMRES
Poschinger et al. (2002)	IDM	Payne
Bourrel and Lesort (2003)	optimal velocity	LWR
Mammar and Haj-Salem (2006)		ARZ
Espié et al. (2006)	ARCHISM	SSMT
El hmam (2006)	generic ABM	LWR, ARZ, Payne

Table 1: Main micro-macro traffic flow models, adapted from El hmam (2006, p. 42)

approach to simulate such systems (Gaud et al., 2008; Gil-Quijano et al., 2010; Gil-Quijano et al., 2012; Picault and Mathieu, 2011; Vo, 2012; Vo et al., 2012a,b). Indeed it offers a large range of techniques to dynamically adapt the level of detail of simulations, couple heterogenous models or detect and reify emergent phenomena (Morvan, 2013).

Thus, in 2013 we started the development of a multi-level agent-based simulator called JAM-FREE within the ISART project. It allows to simulate large road networks efficiently using a dynamic level of detail.

This simulator relies on a multi-level agent-based modeling framework called SIMILAR — formerly IRM4MLS (Morvan and Jolly, 2012; Morvan et al., 2011; Soyeux et al., 2013) — for **S**Imulations with **M**ulti-**I**-**L**evel **A**gents and **R**eactions.

The following sections present our two deliverables: SIMILAR and JAM-FREE.

2 SIMILAR

2.1 Motivations

The simulation of complex system often requires knowledge coming from different sources to obtain relevant results. These sources can either be persons from different application fields, or different viewpoints on the same phenomenon.

Yet, regular multi-agent based simulation meta models lack the structure to manage the aggregation of such systems: their representation of the agents, the environment and the temporal dynamics of the system is designed to support a single viewpoint.

To deal with this issue, we developed a generic approach called SIMILAR to design simulations (fig. 2). This approach relies on a multi-level, influence-reaction and agent-based knowledge representation, more fitting to multiple viewpoints (Ferber and Müller, 1996; Michel, 2007a,b; Morvan and Jolly, 2012; Morvan et al., 2011; Soyeux et al., 2013). The approach includes a generic and modular formal model, a methodology and a simulation API preserving the structure of the formal model (Morvan and Kubera, 2014a,b). Owing to these properties, the design of the above-mentioned simulations is supported during the whole simulation process, is more robust to model revisions and relies on a structure fit to represent the intrinsic complexity of the simulated phenomena.

2.2 Related works

Many meta-models and simulation engines dedicated to multi-level agent-based modeling have been proposed in the literature. We reviewed them in Morvan (2013). One can note that these works generally impose constraints on multi-level models, e.g., on interactions or influence graph structure, while SIMILAR



Figure 2: SIMILAR logo

allows to model and simulate any multi-level situation as it relies on a formal interaction model (Ferber and Müller, 1996).

2.3 SIMILAR micro kernel

SIMILAR is based on the concept of micro kernel. The micro kernel of SIMILAR defines the core classes of the SIMILAR API. It provides a direct correspondence between the concepts developed in the SIMILAR theory and the concrete implementation of simulation. It provides only the bare minimal implementation of these concepts, to leave the implementation of simulation opened to many optimizations. Thus, this kernel is the most appropriate for developers wishing to tune precisely the low level implementation of the simulation.

The micro kernel of SIMILAR is characterized by three components (fig. 3):

- The API of the micro kernel, containing the java classes of that kernel.
- The common libraries of the minimal kernel, containing generic implementations of simulation algorithms and results exporting features.
- The examples illustrating the use of the micro kernel and the common libraries to design simulations.

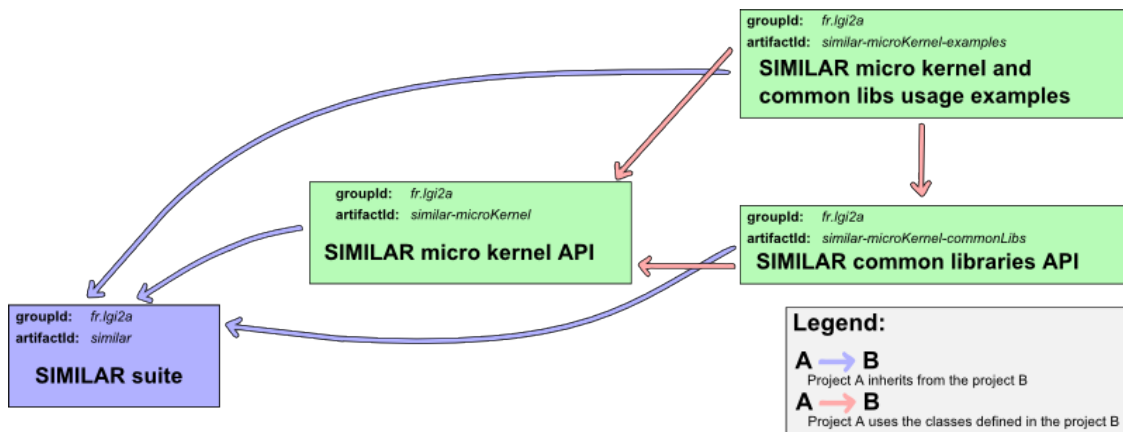


Figure 3: SIMILAR micro kernel structure

2.4 SIMILAR architecture

SIMILAR is designed to have a highly customizable architecture, while making the structure of a simulation as explicit as possible. It distinguishes explicitly elements that are often left to the developers and embedded into the code of the simulations:

The concepts of SIMILAR are embodied as concrete classes (fig. 4):

- The simulation model, containing the declarative part of the simulation (the description of the simulated phenomenon);
- The simulation engine, containing the procedural part of the simulation (the generic algorithms running simulations);
- The observation probes, reading information about the simulation and exporting them in various formats.

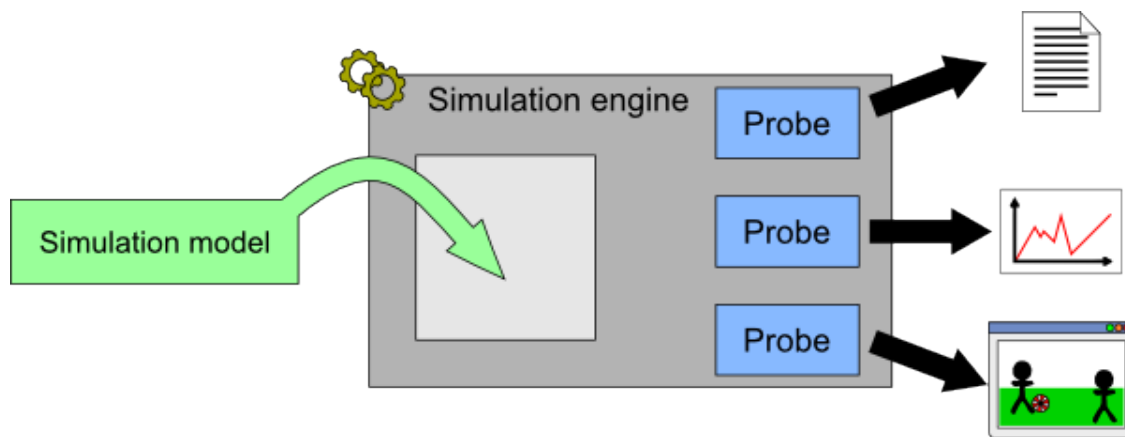


Figure 4: SIMILAR architecture

2.5 Simulation engine

In SIMILAR, the notion of simulation model is separated from the notion of simulation engine to separate the declarative knowledge of the simulation from the procedural knowledge of the simulation.

- A simulation model assembles the simulation-case specific knowledge. It contains the definition of the levels, the agents, the natural action of the environment and the reaction of each level. This knowledge is domain specific.
- A simulation engine assembles the execution-related information of a simulation. It defines how time moves, when to ask the agents to perceive, memorize or decide, when the environment produces its natural action or when the reaction is performed. It is also responsible for the observation and the exporting of the simulation data and responsible for the reaction to the so called system influences.

The separation between model and engine facilitates the optimization of simulations, by using the most appropriate execution mode depending on the simulation: a simulation engine can be implemented using different inner mechanisms to manage the execution of a simulation.

The simulation engine has different roles in the simulation:

- Move the simulation through time and call when it is appropriate:
 - The perception, memorization, decision phases of the agents;
 - The natural phase of the environment;
 - The reaction phase of the levels;
- Ensure that the time-related constraints of the model remain valid during the whole execution of the simulation. Morvan et al. (2011) describes the time constraints that the perception, memorization, natural, decision and reaction phases of the simulation have to verify;
- Ensure that the observation probes are updated appropriately;
- Provide a reaction to the system influences of any simulation.

Considering the simulation engine as a top-class abstraction allows to easily simulate a same model on different computing architectures such as multicore CPUs or GPGPU (general purpose graphical processing unit) as shown on fig. 5.

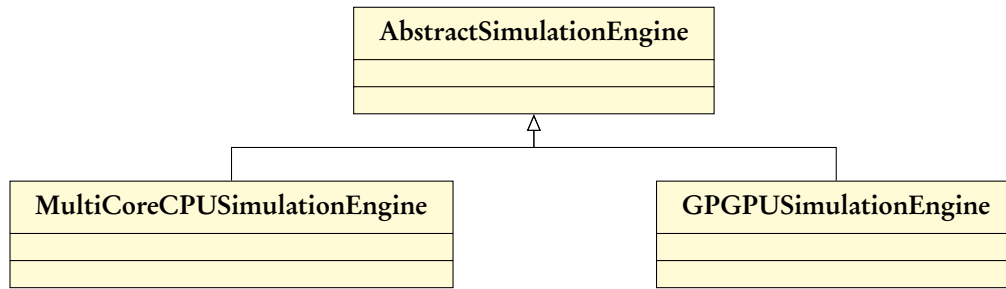


Figure 5: Simulation engines in SIMILAR

2.6 Probes

The observation of the data of the simulation is managed using probes. Probes are objects listening to the evolution of the state of the simulation. Since the state of the simulation is not always consistent (especially during the computation of the reaction), the probes do not decide by themselves when to observe the simulation. Instead, they are registered to the simulation engine. The simulation engine is responsible to tell probes when the state of the simulation is consistent, and thus when probes can observe, process and export information about the simulation. The moments when the state of the simulation is consistent are:

- After the initialization phase of the simulation, but before the execution of the operations related to the first time stamp of the simulation.
- After the execution of a time stamp of the simulation (i.e. after the computation of the reaction leading to the update of the current time stamp of the simulation)
- After the execution of the final time stamp of the simulation.

Since simulations do not always execute peacefully, probes also have to manage the case when the simulation fails because of an error. Moreover, since some probes might use external resources, they also have to be notified when a new simulation will start. Consequently, probes are also notified when:

- Each time a new simulation is performed. In reaction, they will be able for instance to open a writing stream to another file, read another configuration file, etc.
- Each time the simulation stops because of an error. In reaction, they can print the error that caused the simulation to abort, or they can close the streams they were using.

3 JAM-FREE

In this section we present a multi-level agent-based traffic simulator called JAM-FREE (Java, Agent and Multi-level based Framework for Road-traffic Examination and Enhancement).

3.1 Motivations

Our motivation to develop JAM-FREE is to

1. simulate the road traffic on very large zones including both city and highways while
2. being able to test different signing or traffic redirection strategies,
3. assess their precise influence on the traffic flow and
4. understand why traffic perturbations do occur.

The first goal is easily achieved using macroscopic simulation models at the expense of the fourth goal. Conversely, the fourth goal is easily achieved using microscopic simulation models at the expense of the first goal. To deal with this paradox, we designed an dynamic hybrid model where we benefit from the both approaches.

The advantages of this hybrid approach include the ability:

- To obtain both quantitative and qualitative information about the road traffic, using respectively macroscopic and microscopic simulation representations in the same simulation.
- To switch between these representations locally depending:
 - On the simulation needs. For instance understanding the source of a traffic jam.
 - On the computation constraints. For instance managing the CPU load.
- To experiment both macroscopic and microscopic routing strategies, i.e. road load balancing strategies.

3.2 Related works

To the best of our knowledge, the only other work describing a dynamic hybrid model is (Sewall et al., 2011)⁵. However, authors focus on the visualization of data rather than accurate simulation of traffic flow.

⁵Authors maintain a webpage dedicated to this paper: http://gamma.cs.unc.edu/HYBRID_TRAFFIC/

3.3 JAM-FREE use cases

In this section we present some of the JAM-FREE use cases.

The hybrid model can change the traffic representation of a portion of the road network dynamically. This feature can be used to switch from:

- a microscopic representation to a macroscopic representation when the CPU is overused in order to reduce its load;
- a macroscopic representation to a microscopic representation to find out the reason why the traffic jam appeared.

These features can be used jointly to balance the load between two clusters.

The clusters used in the hybrid model are not static. Their number can be increased by dividing existing clusters into two or more clusters. This feature can be used to locate traffic jams. Indeed, macroscopic representations provide quantitative information concerning the traffic, including the mean speed and the free driving speed. A significant difference between these values indicates the presence of a traffic jam. Finding the area where the traffic jam is located consists in dividing the clusters and focusing on the ones where the mean speed is significantly lower than the free driving speed.

3.4 Road network structure

We consider that the road network can be divided dynamically into subsets called clusters (see fig. 6). The traffic is simulated on each cluster using various heterogeneous models depending on the situation: either a microscopic model or a macroscopic model (see fig. 7).

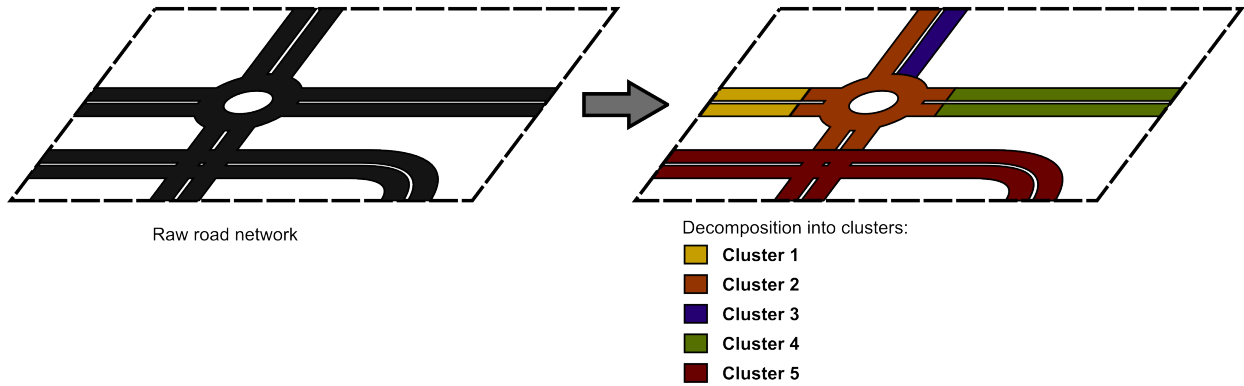


Figure 6: SIMILAR road network structure

For computation time efficiency reasons, we choose not to model the road network at a physical level. Indeed, such a model requires the vehicles to interpret a large continuous zone of asphalt with blank lines as separated lanes. Such computations are unnecessary complex, since in our use case (France) rational drivers usually never drive over the blank lines separating the lanes if they are not overtaking. To keep the model simple, the road network is only modeled at a semantic level.

The road network is modeled as a set of interconnected roads. Each road contains a set of lanes (usually from 1 to 5). Roads also contain vertical signs (e.g. stop sign, speed limit sign) that are bond either to all the lanes or to specific lanes (e.g. extraction lane in the highway, slow vehicles lane). In our model, we support a specific subset of the french vertical signs . The connection points of the roads are called nodes. We distinguish different types of nodes:

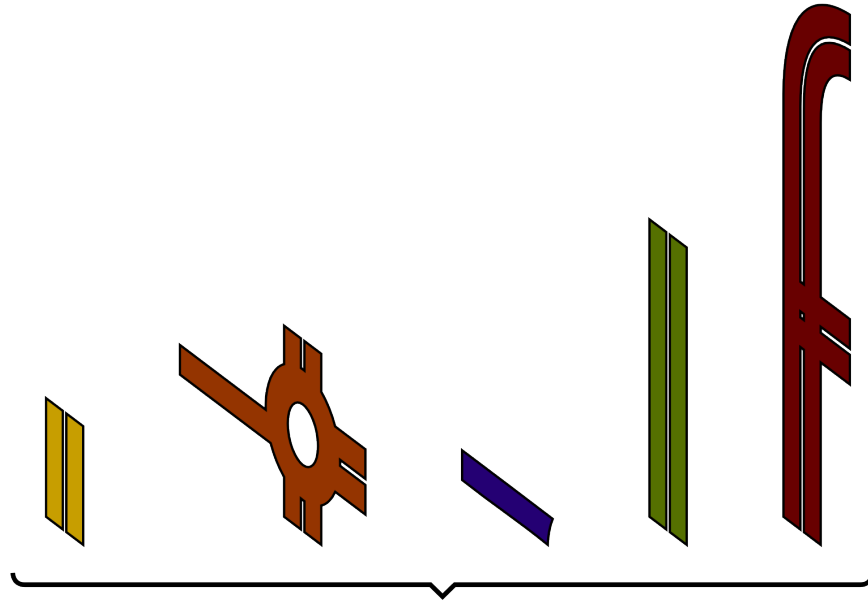
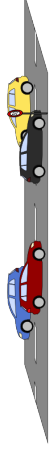


Figure 7: SIMILAR traffic representations

Cluster 1: simulation with
macroscopic traffic flow model

$$\begin{aligned} \frac{(\partial \rho)}{(\partial t)} + \frac{(\partial q)}{(\partial x)} &= 0 \quad (1) \\ \frac{(\partial v)}{(\partial t)} + v \frac{(\partial \rho)}{(\partial x)} &= \frac{(V(\rho) - v)}{(\tau)} - \frac{C_0^2(\partial \rho)}{\rho \frac{(\partial x)}{(\partial x^2)}} + \frac{f(\partial^2 v)}{\rho \frac{(\partial x^2)}{(\partial x^2)}} \quad (2) \end{aligned}$$

Cluster 2: simulation with
microscopic multi-agent system



Cluster 3: simulation with
macroscopic traffic flow model

$$\begin{aligned} \frac{(\partial \rho)}{(\partial t)} + \frac{(\partial q)}{(\partial x)} &= 0 \quad (1) \\ \frac{(\partial v)}{(\partial t)} + v \frac{(\partial \rho)}{(\partial x)} &= \frac{(V(\rho) - v)}{(\tau)} - \frac{C_0^2(\partial \rho)}{\rho \frac{(\partial x)}{(\partial x^2)}} + \frac{f(\partial^2 v)}{\rho \frac{(\partial x^2)}{(\partial x^2)}} \quad (2) \end{aligned}$$

Cluster 4: simulation with
macroscopic traffic flow model

$$\begin{aligned} \frac{(\partial \rho)}{(\partial t)} + \frac{(\partial q)}{(\partial x)} &= 0 \quad (1) \\ \frac{(\partial v)}{(\partial t)} + v \frac{(\partial \rho)}{(\partial x)} &= \frac{(V(\rho) - v)}{(\tau)} - \frac{C_0^2(\partial \rho)}{\rho \frac{(\partial x)}{(\partial x^2)}} + \frac{f(\partial^2 v)}{\rho \frac{(\partial x^2)}{(\partial x^2)}} \quad (2) \end{aligned}$$

Cluster 5: simulation with
microscopic traffic flow model

$$\begin{aligned} \tilde{x}_\alpha(t) &= \tilde{v}_\alpha(t) = F(v_\alpha(t), s_\alpha(t), v_{\alpha-1}(t)) \\ \dot{\tilde{v}}_\alpha(t) &= f(x_\alpha(t), v_\alpha(t), x_{\alpha-1}(t), v_{\alpha-1}(t), \dots, x_{\alpha-n_\alpha}(t), v_{\alpha-n_\alpha}(t)) \\ \tilde{v}_\alpha^{t+1} &= f(\tilde{v}_\alpha^t, \tilde{v}_{\alpha-1}^t, \dots) \end{aligned}$$

- Crossroads nodes
- Roundabout nodes
- Highway insertion node
- Highway extraction node

The naming and identification of road network components follow the S etra⁶ specification⁷ (S etra, 2010).

3.5 Vehicle behaviors

The behavior of a vehicle in our simulation is designed to be modular, customizable and easily extendible. To achieve these properties, the behavior of a vehicle relies on the following "chain of responsibility" design pattern (see fig. 8).

The behavior of a vehicle can be seen as the sum of three different parts:

- A navigation behavior when the current lane of the vehicle does not lead to the desired destination of the vehicle. This behavior consists in changing the lane of the vehicle until the current lane of the vehicle leads to the desired destination.
- An overtaking behavior when the vehicle either wants to increase its speed by changing its lane, or when the vehicle has to swerve because a faster vehicle is tailing it.
- A behavior when no lane change is required (acceleration model).

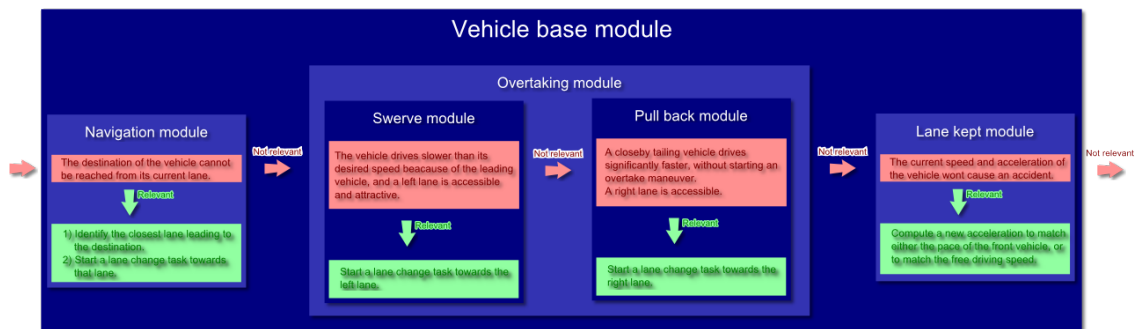


Figure 8: Vehicle behavior structure in SIMILAR

⁶S etra (Service d etudes sur les transports, les routes et leurs am enagements) is a technical service of the Minist ere de l Ecologie, du D veloppement Durable et de l Energie dedicated to transportation issues: <http://www.setra.developpement-durable.gouv.fr>

⁷<http://dtrf.setra.fr/pdf/pj/Dtrf/0005/Dtrf-0005792/DT5792.pdf>

3.5.1 Acceleration model

JAM-FREE implements the highly used Intelligent-Driver Model (IDM) to model the acceleration behavior of vehicles (Kesting et al., 2010).

The IDM is a microscopic traffic flow model, i.e., each vehicle-driver combination constitutes an active "particle" in the simulation. Such model characterize the traffic state at any given time by the positions and speeds of all simulated vehicles. In case of multi-lane traffic, the lane index complements the state description. More specifically, the IDM is a car-following model. In such models, the decision of any driver to accelerate or to brake depends only on his or her own speed, and on the position and speed of the "leading vehicle" immediately ahead. Lane-changing decisions, however, depend on all neighboring vehicles (see the lane-changing model MOBIL). The model structure of the IDM can be described as follows:

- The influencing factors (model input) are the own speed v , the bumper-to-bumper gap s to the leading vehicle, and the relative speed (speed difference) of the two vehicles (positive when approaching).
- The model output is the acceleration chosen by the driver for this situation.
- The model parameters describe the driving style, i.e., whether the simulated driver drives slow or fast, careful or reckless

3.5.2 Lane changing model

Lane changes takes place, if:

- the potential new target lane is more attractive, i.e., the "incentive criterion" is satisfied,
- and the change can be performed safely, i.e., the "safety criterion" is satisfied.

JAM-FREE implements a modified version of the lane changing model MOBIL⁸.

Chosen acceleration and lane changing models are presented in detail in the JAM-FREE documentation.

3.6 Traffic generation

In our model, each Traffic start connector (see this page) is a special connector that will generate traffic on the road it is attached to. Note that such a connector has to be created and put on each lane where the traffic appears. The creation of vehicles is managed by a traffic input point agent.

Various implementation of this agent currently exist:

- "Flow-mass traffic input point " agents, generating the traffic flow using a flow-mass parameter
- "Scripted traffic input point " agents, generating the traffic flow using user-defined events. The created vehicles and the creation dates are manually specified by the users.

⁸<http://xxx.uni-augsburg.de/abs/cond-mat/0002177>

3.7 User interfaces

JAM-FREE simulations and models can be created using a convenient hybrid XML/Java system:

- simple simulations can be created using only XML files,
- complex simulations (for instance with dedicated vehicle generation methods, or behaviors) can be created by implementing Java classes that are referred in the XML files.

The XML files describe the "physical" road network as well as the simulation itself (traffic generation methods and models). An example of simulation is shown in appendix A.

To simply manage simulations, a graphical user interface called JFF (JAM-FREE Framework) has been developed (see appendix B).

4 Conclusion

Within the ISART project of CISIT, we developed a multi-level agent-based traffic simulator called JAM-FREE, relying on a dedicated meta-model called SIMILAR. These software programs have been developed using state of the art software engineering tools such as SonarQube⁹ (see appendix C), Maven¹⁰, Jenkins¹¹ or Hamcrest¹². Massive unit tests were carried to verify simulation engines and models. For instance, 545800 tests (265Mo of source code), each describing a specific use case, have been generated to validate the perception model of JAM-FREE.

4.1 Deliverables

The meta-model and simulation engine of SIMILAR have been implemented in Java, and will be available freely in a few weeks under the CeCILL-B¹³ license. The project is currently hosted on <https://forge.univ-artois.fr> with a limited access to the source code.

JAM-FREE is still under development. However, a preliminary demo version can be distributed to the CISIT partners if needed.

Two papers are in preparation:

- Morvan and Kubera (2014b), that describes the SIMILAR meta-model,
- Morvan and Kubera (2014a), that proposes novel ways to manage uncertainty in multi-agent-based simulations.

4.2 Perspectives

During the next phase of CISIT, we will pursue the development of JAM-FREE. For instance, one of our goal is to be able to import data from the open source geographical information system OpenStreetMap¹⁴ and directly use it in JAM-FREE.

⁹<http://www.sonarqube.org>

¹⁰<http://maven.apache.org>

¹¹jenkins-ci.org

¹²<https://code.google.com/p/hamcrest/>

¹³http://cecill.info/licences/Licence_CeCILL-B_V1-en.html

¹⁴<http://www.openstreetmap.org>

Appendices

A JAM-FREE XML files

A minimal JAM-FREE simulation is defined by a set of XML files. A simple example is given in the following. The main file specifies simulation's main options as well as the XML files describing the road network structure and traffic generation.

Listing 1: JAM-FREE: main XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<jamFreeModel>

  <!-- Identify the file where the infrastructure of the road network is described. -->
  <roadNetwork fileName="infrastructure/navigation-infrastructure.xml" />

  <!-- Describe how time moves through simulation -->
  <time sampling="0.05"> <!-- A time step occurs every 0.05 second of the simulated world (optional). -->
    <simulationEnd time="180" /> <!-- The simulation runs during 60 seconds. -->
    <microscopicTrafficEvolution period="1" /> <!-- The "microscopic - traffic" level acts at each time step (optional) -->
  </time>

  <!-- Define the initial state of the different levels of the simulation. -->
  <levels>
    <!-- Identify the file where the initial state of the "microscopic - traffic" level is described. -->
    <microscopicTrafficLevel fileName="microscopicLevel/microscopicLevel-example1.xml" />
  </levels>

  <!-- Define how the results of the simulation can be seen. -->
  <probes />

</jamFreeModel>
```

The infrastructure files describes the road components contained in the road network and their connections.

Listing 2: JAM-FREE: infrastructure XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<roadnetwork>
  <!--
    Description of the road components contained in the road network.
  -->
  <road id="N9PR1_section1" numberOfLanes="2" width="4" semanticLength="1000">
    <type value="national"/>
    <input plo="N9PR1" distanceFromPlo="0">
      <laneIdRange from="0" to="1"/>
    </input>
    <output plo="N9PR1" distanceFromPlo="1000">
      <laneIdRange from="0" to="1"/>
    </output>
    <shape>
      <point x="0" y="8"/>
      <point x="1000" y="8"/>
    </shape>
  </road>
  <road id="N9PR1_section2" numberOfLanes="2" width="4" semanticLength="500">
    <type value="national"/>
    <input plo="N9PR1" distanceFromPlo="1000">
      <laneIdRange from="0" to="1"/>
    </input>
    <output plo="N9PR1" distanceFromPlo="1500">
      <laneIdRange from="0" to="1"/>
    </output>
    <reachableDestinations>
      <laneDestination>
        <laneId id="1" />
        <destination name="Paris" />
      </laneDestination>
      <laneDestination>
        <laneId id="0" />
        <destination name="Dunkerque" />
        <destination name="Arras" />
      </laneDestination>
    </reachableDestinations>
    <shape>
      <point x="1000" y="8"/>
      <point x="1500" y="8"/>
    </shape>
  </road>
  <road id="D19PR1_section1" numberOfLanes="1" width="4" semanticLength="300">
    <type value="national"/>
```

```

        <input plo="N9PR1" distanceFromPlo="1500">
          <laneId id="1"/>
        </input>
        <output plo="D19PR1" distanceFromPlo="300">
          <laneId id="0"/>
        </output>
        <reachableDestinations>
          <laneDestination>
            <laneId id="1" />
            <destination name="Paris" />
          </laneDestination>
        </reachableDestinations>
        <shape>
          <point x="1500" y="4"/>
          <point x="1800" y="4"/>
        </shape>
      </road>
      <road id="D20PR1_section1" numberOfLanes="1" width="4" semanticLength="300">
        <type value="national"/>
        <input plo="N9PR1" distanceFromPlo="1500">
          <laneId id="0"/>
        </input>
        <output plo="D20PR1" distanceFromPlo="300">
          <laneId id="0"/>
        </output>
        <reachableDestinations>
          <laneDestination>
            <laneId id="0" />
            <destination name="Dunkerque" />
            <destination name="Arras" />
          </laneDestination>
        </reachableDestinations>
        <shape>
          <point x="1500" y="8"/>
          <point x="1501" y="8"/>
          <point x="1651" y="158"/>
        </shape>
      </road>

<!--
--> Description of the connection between the road components.

<connection outputComponentId="N9PR1_section1" inputComponentId="N9PR1_section2">
  <input plo="N9PR1" distanceFromPlo="1000">
    <laneIdRange from="0" to="1"/>
  </input>
  <output plo="N9PR1" distanceFromPlo="1000">
    <laneIdRange from="0" to="1"/>
  </output>
</connection>
<connection outputComponentId="N9PR1_section2" inputComponentId="D19PR1_section1">
  <input plo="N9PR1" distanceFromPlo="1500">
    <laneId id="1"/>
  </input>
  <output plo="N9PR1" distanceFromPlo="1500">
    <laneId id="1"/>
  </output>
</connection>
<connection outputComponentId="N9PR1_section2" inputComponentId="D20PR1_section1">
  <input plo="N9PR1" distanceFromPlo="1500">
    <laneId id="0"/>
  </input>
  <output plo="N9PR1" distanceFromPlo="1500">
    <laneId id="0"/>
  </output>
</connection>

<road id="D19PR1_section2" numberOfLanes="2" width="4" semanticLength="500">
  <type value="national"/>
  <input plo="D19PR1" distanceFromPlo="300">
    <laneIdRange from="0" to="1" />
  </input>
  <output plo="D19PR1" distanceFromPlo="800">
    <laneIdRange from="0" to="1" />
  </output>
  <explicitSpecification>
    <!-- We tell that the lane having input point "0" starts at the beginning of the road. -->
    <laneStartingPoint inputId="0" />
  </explicitSpecification>
  <shape>
    <point x="1800" y="8"/>
    <point x="2300" y="8"/>
  </shape>
</road>
<road id="D19PR1_section3" numberOfLanes="2" width="4" semanticLength="500">
  <type value="national"/>
  <input plo="D19PR1" distanceFromPlo="800">
    <laneIdRange from="0" to="1" />
  </input>
  <output plo="D19PR1" distanceFromPlo="1300">
    <laneIdRange from="0" to="1" />
  </output>

```

```

        <reachableDestinations>
            <laneDestination>
                <laneId id="1" />
                <destination name="Paris" />
            </laneDestination>
            <laneDestination>
                <laneId id="0" />
                <destination name="Lille" />
            </laneDestination>
        </reachableDestinations>
    </shape>
    <point x="2300" y="8"/>
    <point x="2800" y="8"/>
</shape>
</road>
<connection outputComponentId="D19PR1_section1" inputComponentId="D19PR1_section2">
    <input plo="D19PR1" distanceFromPlo="300">
        <laneId id="1"/>
    </input>
    <output plo="D19PR1" distanceFromPlo="300">
        <laneId id="0"/>
    </output>
</connection>
<connection outputComponentId="D19PR1_section2" inputComponentId="D19PR1_section3">
    <input plo="D19PR1" distanceFromPlo="800">
        <laneIdRange from="0" to="1"/>
    </input>
    <output plo="D19PR1" distanceFromPlo="800">
        <laneIdRange from="0" to="1"/>
    </output>
</connection>

<road id="D25PR1_section1" numberOfLanes="1" width="4" semanticLength="300">
    <type value="national"/>
    <input plo="D19PR1" distanceFromPlo="1300">
        <laneId id="0"/>
    </input>
    <output plo="D25PR1" distanceFromPlo="300">
        <laneId id="0"/>
    </output>
    <reachableDestinations>
        <laneDestination>
            <laneId id="0" />
            <destination name="Lille" />
        </laneDestination>
    </reachableDestinations>
    <shape>
        <point x="2800" y="8"/>
        <point x="2802" y="8"/>
        <point x="2802" y="306"/>
    </shape>
</road>
<connection outputComponentId="D19PR1_section3" inputComponentId="D25PR1_section1">
    <input plo="D19PR1" distanceFromPlo="1300">
        <laneId id="0"/>
    </input>
    <output plo="D19PR1" distanceFromPlo="1300">
        <laneId id="0"/>
    </output>
</connection>
<road id="D26PR1_section1" numberOfLanes="1" width="4" semanticLength="300">
    <type value="national"/>
    <input plo="D19PR1" distanceFromPlo="1300">
        <laneId id="1"/>
    </input>
    <output plo="D26PR1" distanceFromPlo="300">
        <laneId id="0"/>
    </output>
    <reachableDestinations>
        <laneDestination>
            <laneId id="1" />
            <destination name="Paris" />
        </laneDestination>
    </reachableDestinations>
    <shape>
        <point x="2800" y="4"/>
        <point x="3100" y="4"/>
    </shape>
</road>
<connection outputComponentId="D19PR1_section3" inputComponentId="D26PR1_section1">
    <input plo="D19PR1" distanceFromPlo="1300">
        <laneId id="1"/>
    </input>
    <output plo="D19PR1" distanceFromPlo="1300">
        <laneId id="1"/>
    </output>
</connection>
</roadnetwork>

```


The microscopic level files defines vehicle generation and destruction points, *i.e.*, the boundaries of the microscopic representation.

Listing 3: JAM-FREE: microscopic level XML file

```
<?xml version="1.0" encoding="UTF-8"?>

<microTrafficLevel>
<!--
--> Create the generation points at the beginning of the simulated A1 highway (at the PLO A1PR35D)
-->
    <trafficGenerationPoint>
        <location>
            <position roadId="N9PR1_section1" plo="N9PR1" distanceFromPlo="0" laneId="0" />
        </location>
        <behavior>
            <rhythmModel
                javaClass="fr.lgi2a.jamFree.libraries.microscopicLevel.generationRythmModels.FlowMassBased_RhythmModel"
                parametersFile="microscopicLevel/generationPoints/A1PR35D_section1-rhythmParameters.xml"
            />
            <vehicleGenerationModel
                javaClass="fr.lgi2a.jamFree.libraries.microscopicLevel.vehicleGenerationModels.
                    UniformWithIDMForwardAcceleration_WithMobilLaneChange_SpecificDestination_GenerationModel"
                parametersFile="microscopicLevel/generationPoints/A1PR35D_section1-generationParameters.xml"
            />
        </behavior>
    </trafficGenerationPoint>

<!--
--> Create the destruction point at the end of the simulated A1 national road (at the PLO A1PR38D)
-->
    <trafficDestructionPoint>
        <location>
            <position roadId="D20PR1_section1" plo="D20PR1" distanceFromPlo="300" laneId="0" />
        </location>
    </trafficDestructionPoint>
    <trafficDestructionPoint>
        <location>
            <position roadId="D26PR1_section1" plo="D26PR1" distanceFromPlo="300" laneId="0" />
        </location>
    </trafficDestructionPoint>
    <trafficDestructionPoint>
        <location>
            <position roadId="D25PR1_section1" plo="D25PR1" distanceFromPlo="300" laneId="0" />
        </location>
    </trafficDestructionPoint>
</microTrafficLevel>
```

Finally, each generation point is characterized by two XML files specifying how (generation parameters) and when (rhythm parameters) vehicles are created.

Listing 4: JAM-FREE: generation parameters XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="headwayTimeUsedInGenerationValidation">1.5</entry>
    <entry key="vehicleMinimalGapUsedInGenerationValidation">2.0</entry>

    <entry key="vehicleLength">4.13</entry>
    <entry key="accelerationModel_initialSpeed">20.0</entry>

    <entry key="accelerationModel_maximalAcceleration">1.962</entry>
    <entry key="accelerationModel_maximalDeceleration">5.886</entry>
    <entry key="accelerationModel_maximalSpeed">36.1111</entry>
    <entry key="accelerationModel_minimalVehicleGap">2.0</entry>
    <entry key="accelerationModel_headwayTime">1.5</entry>

    <entry key="lanceChange_accelerationAnticipationModel_maximalAcceleration">0.2</entry>
    <entry key="lanceChange_accelerationAnticipationModel_maximalDeceleration">3.0</entry>
    <entry key="lanceChange_accelerationAnticipationModel_headwayTime">1.4</entry>
    <entry key="lanceChange_accelerationAnticipationModel_minimalVehicleGap">2.5</entry>

    <entry key="lanceChange_manueverDuration">2.5</entry>
    <entry key="lanceChange_maximumSafeDeceleration">4.0</entry>
    <entry key="lanceChange_politenessFactor">0.35</entry>
    <entry key="lanceChange_accelerationThreshold">0.2</entry>
    <entry key="lanceChange_rightLaneAccelerationBias">0.1</entry>

    <entry key="lanceChange_thresholdDistanceComputation_headwayTime">1.5</entry>
    <entry key="lanceChange_thresholdDistanceComputation_minimalVehicleGap">2.0</entry>

    <entry key="checkPoints">Paris</entry>
</properties>
```

Listing 5: JAM-FREE: generation rhythm XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="flowMass">0.20</entry>
</properties>
```

B JAM-FREE Framework

JAM-FREE Framework (JFF) is a graphical user interface that allow users to simply manage simulations (load, run, stop, analyse results). Screenshots of JFF are shown in figures 9–12.

C SonarQube screenshots

SonarQube is an open source platform for Continuous Inspection (CI) of code quality. It computes many interesting quality metrics such as the percentage of cases covered by unit tests. As the fig. 13 and 14 show, SIMILAR obtains excellent reports on SonarQube.

References

- Bourrel, E. and Lesort, J. (2003). Mixing micro and macro representations of traffic flow: a hybrid model based on the LWR theory. *82th Annual Meeting of the Transportation Research Board*.
- El hmam, M. (2006). *Contribution à la modélisation et à la simulation hybride du flux de trafic*. PhD thesis, Université d’Artois.
- Espié, S., Gattuso, D., and Galante, F. (2006). Hybrid traffic model coupling macro- and behavioral microsimulation. In *85th Annual Meeting of Transportation Research Board*, Washington D.C.
- Ferber, J. and Müller, J.-P. (1996). Influences and reaction: a model of situated multiagent systems. In *2nd International Conference on Multi-agent systems (ICMAS’96)*, pages 72–79.
- Gaud, N., Galland, S., Gechter, F., Hilaire, V., and Koukam, A. (2008). Holonic multilevel simulation of complex systems : Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, 16:1659–1676.
- Gil-Quijano, J., Hutzler, G., and Louail, T. (2010). Accroche-toi au niveau, j’enlève l’échelle: Éléments d’analyse des aspects multiniveaux dans la simulation à base d’agents. *Revue d’Intelligence Artificielle*, 24(5):625–648.
- Gil-Quijano, J., Louail, T., and Hutzler, G. (2012). From biological to urban cells: Lessons from three multilevel agent-based models. In Desai, N., Liu, A., and Winikoff, M., editors, *Principles and Practice of Multi-Agent Systems*, volume 7057 of *Lecture Notes in Computer Science*, pages 620–635. Springer.
- Kesting, A., Treiber, M., and Helbing, D. (2010). Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605.
- Magne, L., Rabut, S., and Gabard, J. (2000). Towards an hybrid macro-micro traffic flow simulation model. *Proceedings of the INFORMS Salt Lake City String 2000 Conference*.
- Mammar, S. and Lebacque, J. and Haj-Salem, H. (2006). Hybrid model based on second-order traffic model. *85th Annual Meeting of Transportation Research Board*, 1(06-2160).

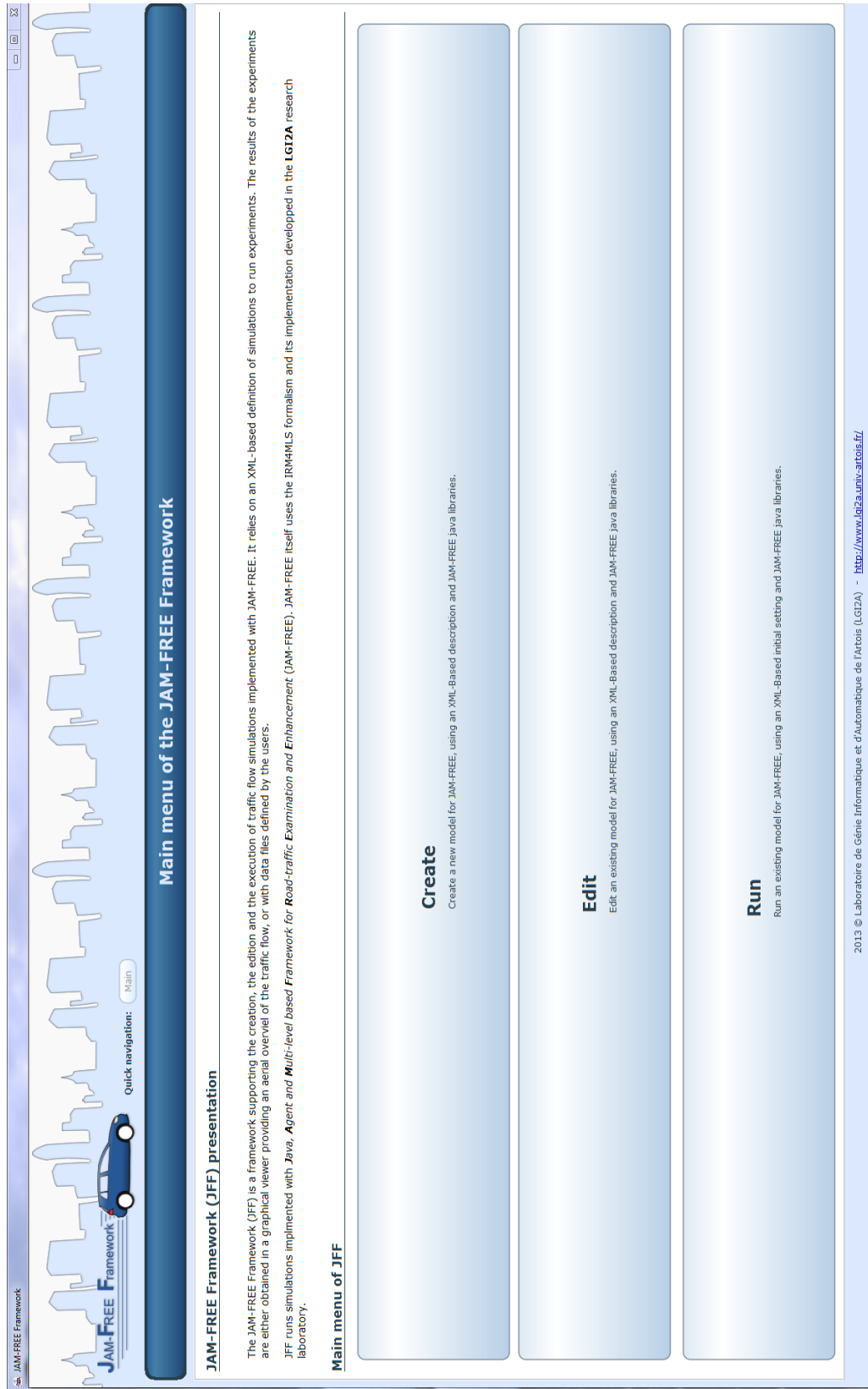


Figure 9: Screenshot of JFF: main menu

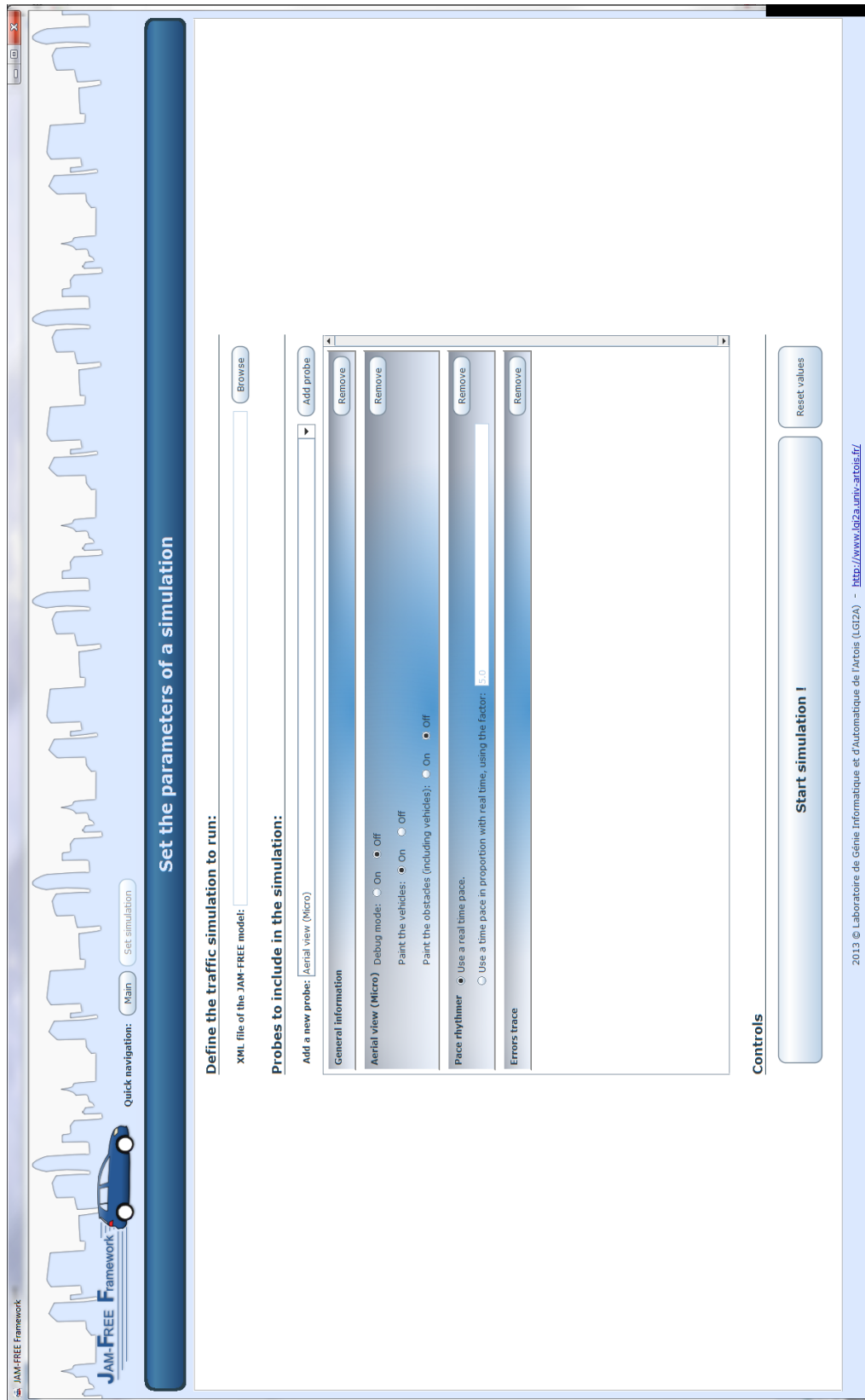


Figure 10: Screenshot of JFF: set the parameters of a simulation

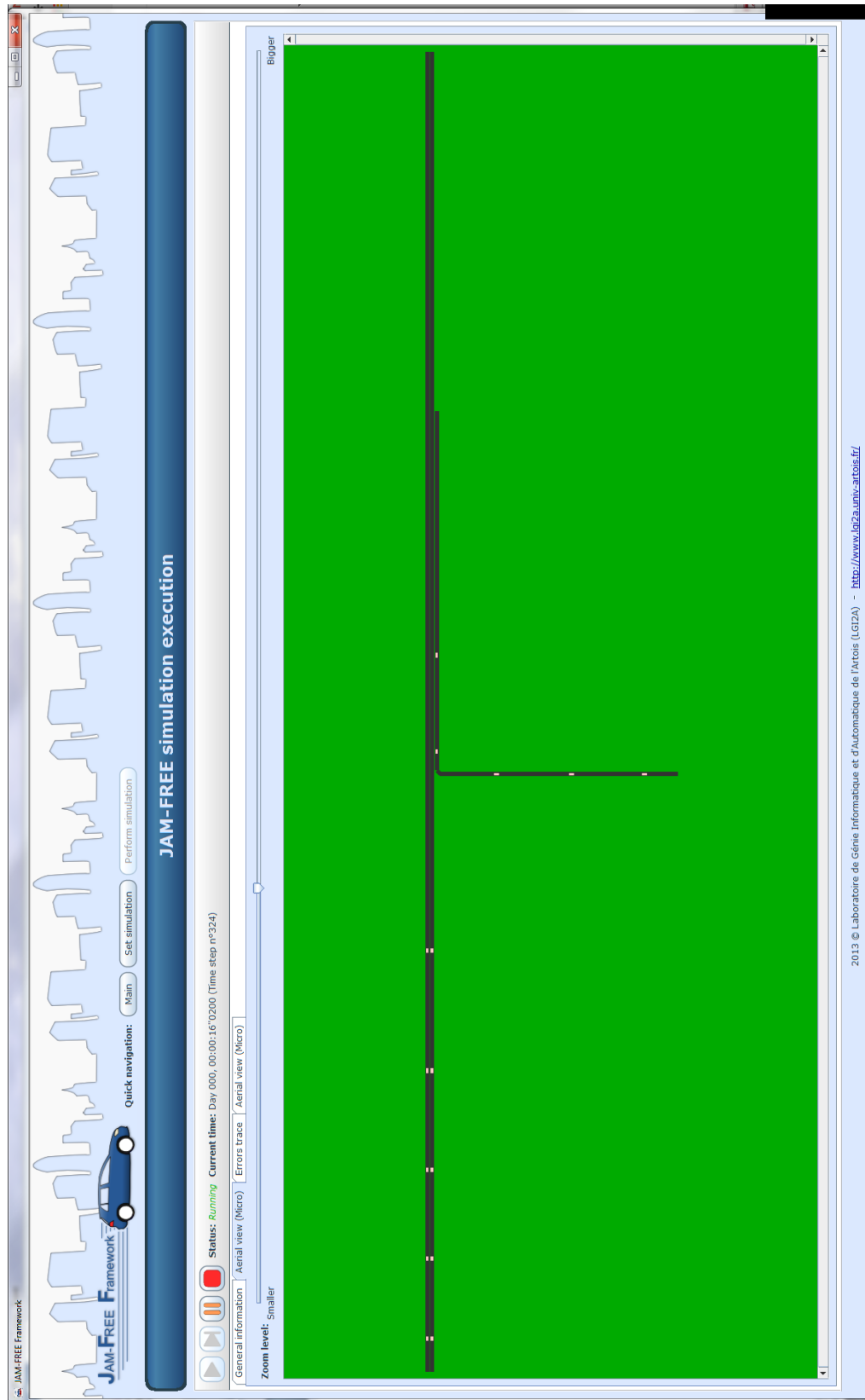


Figure 11: Screenshot of JFF: simulation execution

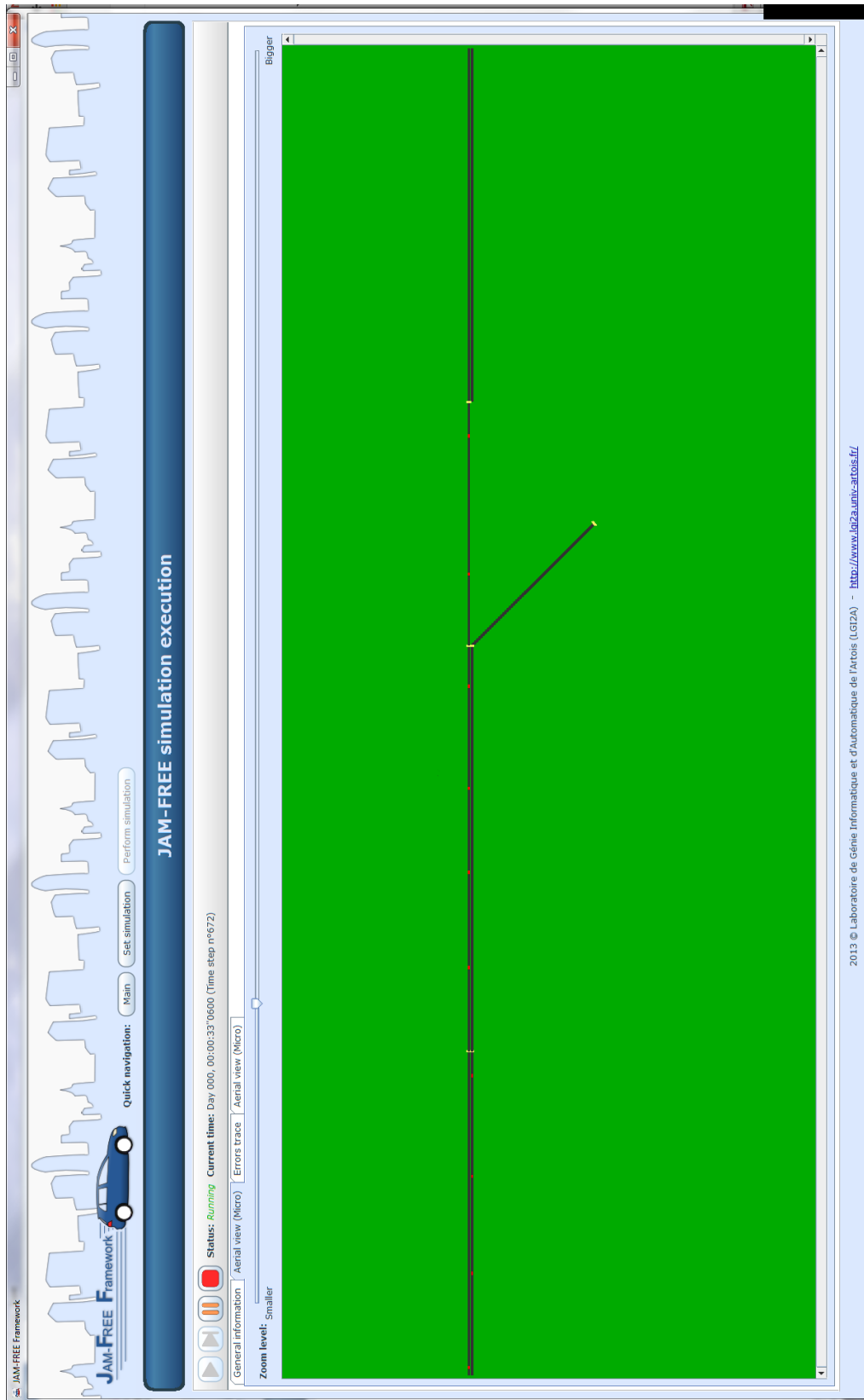


Figure 12: Screenshot of JFF: simulation execution (debug mode)

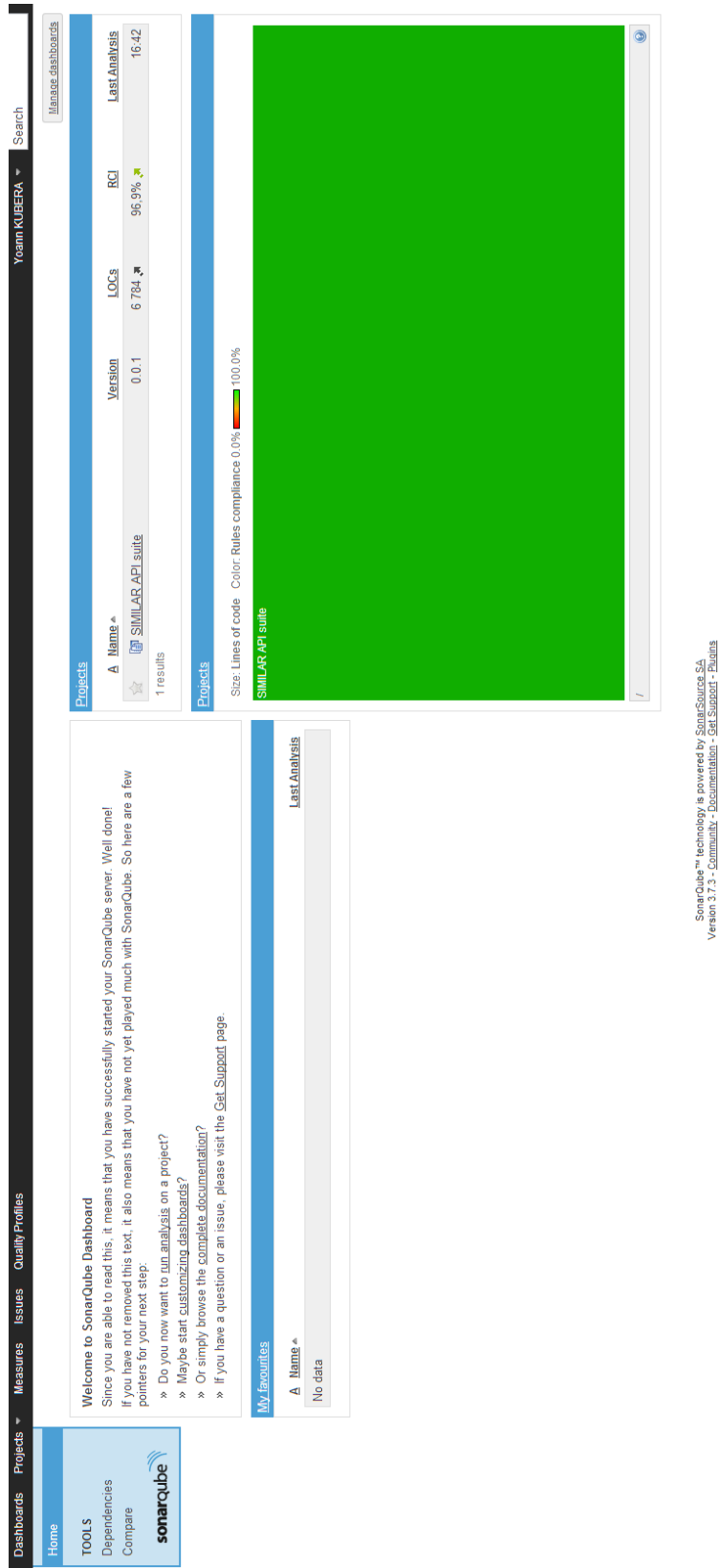


Figure 13: Screenshot of SonarQube: Main screen that shows that SIMILAR compiles at 96.9% to the best coding practices

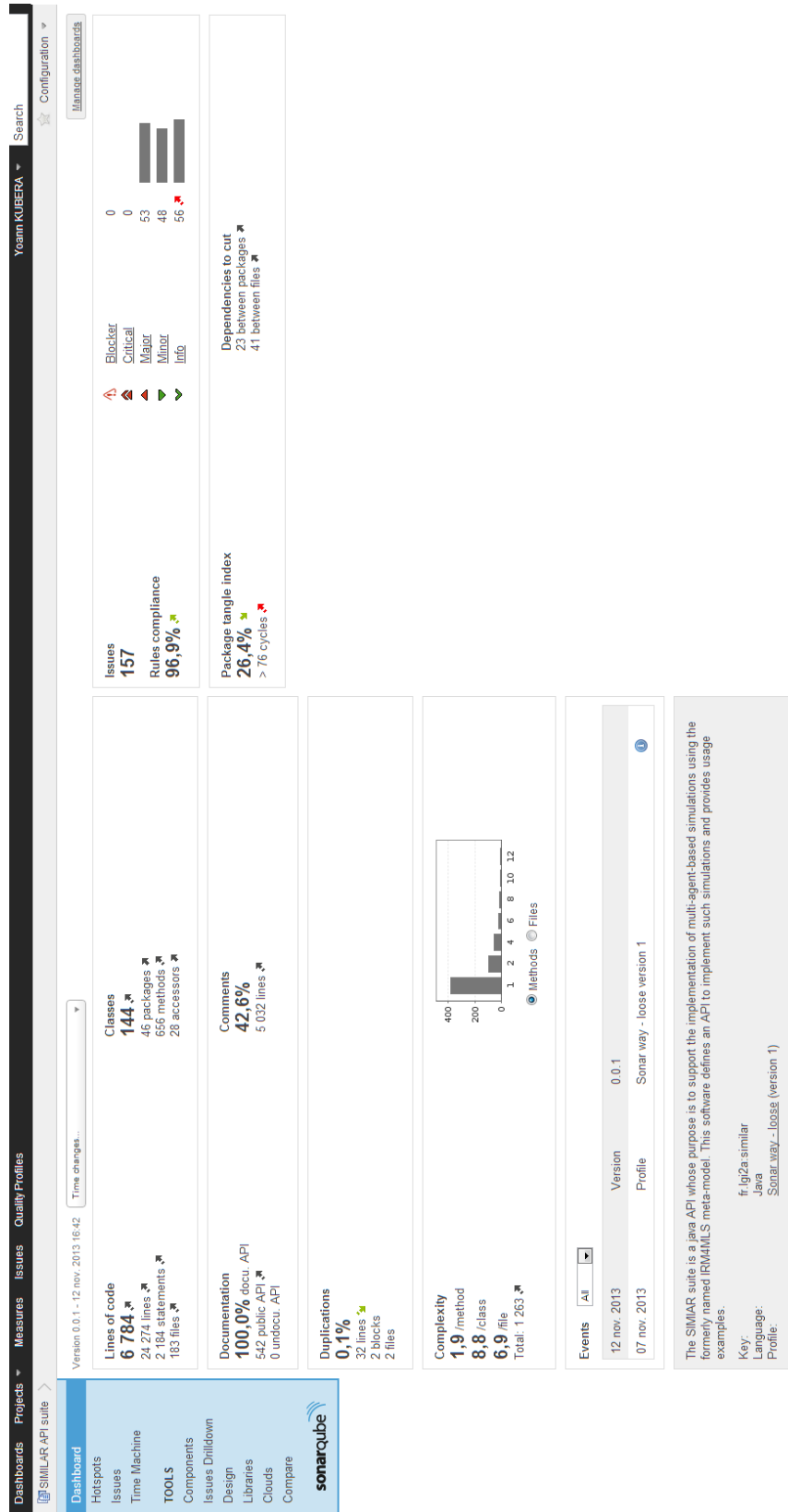


Figure 14: Screenshot of SonarQube: Screen showing the different metrics of SIMILAR

- Michel, F. (2007a). The IRM4S model: the influence/reaction principle for multiagent based simulation. In *Proc. of 6th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 1–3.
- Michel, F. (2007b). Le modèle IRM4S. de l’utilisation des notions d’influence et de réaction pour la simulation de systèmes multi-agents. *Revue d’Intelligence Artificielle*, 21:757–779.
- Morvan, G. (2013). Multi-level agent-based modeling - a literature survey. *CoRR*, abs/1205.0561.
- Morvan, G. and Jolly, D. (2012). Multi-level agent-based modeling with the Influence Reaction principle. *CoRR*, abs/1204.0634.
- Morvan, G. and Kubera, Y. (2014a). On time and consistency in multi-agent-based simulations. Working paper.
- Morvan, G. and Kubera, Y. (2014b). SIMILAR: Simulations with multi-level agents and reactions. Working paper.
- Morvan, G., Veremme, A., and Dupont, D. (2011). IRM4MLS: the influence reaction model for multi-level simulation. In Bosse, T., Geller, A., and Jonker, C., editors, *Multi-Agent-Based Simulation XI*, volume 6532 of *Lecture Notes in Artificial Intelligence*, pages 16–27. Springer.
- Picault, S. and Mathieu, P. (2011). An interaction-oriented model for multi-scale simulation. In Walsh, T., editor, *Twenty-Second International Joint Conference on Artificial Intelligence*, pages 332–337, Barcelona, Spain. AAAI Press.
- Poschinger, A., Kates, R., and Keller, H. (2002). Coupling of concurrent macroscopic and microscopic traffic flow models using hybrid stochastic and deterministic disaggregation. *Transportation and Traffic Theory for the 21st century*.
- Sétra (2010). Identification et localisation sur le reseau routier national. Technical report, Sétra, 110 rue de Paris 77171 Sourdun - France.
- Sewall, J., Wilkie, D., and Lin, M. C. (2011). Interactive hybrid simulation of large-scale traffic. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 30(6).
- Soyez, J.-B., Morvan, G., Dupont, D., and Merzouki, R. (2013). A methodology to engineer and validate dynamic multi-level multi-agent based simulations. In *Multi-Agent-Based Simulation XIII*, volume 7838 of *Lecture Notes in Artificial Intelligence*, pages 130–142. Springer.
- Vo, A. (2012). *An operational architecture to handle multiple levels of representation in agent-based models*. PhD thesis, Université Paris VI.
- Vo, D.-A., Drogoul, A., and Zucker, J.-D. (2012a). An operational meta-model for handling multiple scales in agent-based simulations. In *International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–6, Ho Chi Minh City. IEEE.
- Vo, D.-A., Drogoul, A., Zucker, J.-D., and Ho, T.-V. (2012b). A modelling language to represent and specify emerging structures in agent-based model. In Desai, N., Liu, A., and Winikoff, M., editors, *Principles and Practice of Multi-Agent Systems*, volume 7057 of *Lecture Notes in Computer Science*, pages 212–227. Springer.